Estimating Grammar Parameters using Bounded Memory

Tim Oates[1] and Brent Heeringa[2]

[1] Department of Computer Science and Electrical Engineering. University of Maryland Baltimore County. 1000 Hilltop Circle. Baltimore, MD 21250
oates@eecs.umbc.edu

[2] Department of Computer Science. University of Massachusetts, Amherst. Amherst, MA 01003
heeringa@cs.umass.edu

Abstract. Estimating the parameters of stochastic context-free grammars (SCFGs) from data is an important, well-studied problem. Almost without exception, existing approaches make repeated passes over the training data. The memory requirements of such algorithms are ill-suited for embedded agents exposed to large amounts of training data over long periods of time. We present a novel algorithm, called HOLA, for estimating the parameters of SCFGs that computes summary statistics for each string as it is observed and then discards the string. The memory used by HOLA is bounded by the size of the grammar, not by the amount of training data. Empirical results show that HOLA performs as well as the Inside-Outside algorithm on a variety of standard problems, despite the fact that it has access to much less information.

1 Introduction

Stochastic context-free grammars (SCFGs) are perhaps best known as a tool for expressing the syntactic structure of natural languages. However, their utility extends well beyond this one domain. In recent years SCFGs have been widely applied to problems in computational biology, such as modeling the secondary structure of RNA families [1]. Other applications include visual recognition of activities and language modeling for speech recognition [2].

A problem of central importance in each of these applications is inducing SCFGs from data. Solutions to this problem almost always have the following two properties: (1) they make multiple passes through the data, often expending significant computation during each pass and (2) they require large amounts of data to accurately estimate production probabilities. One experiment reported in the literature used the 30 million word Wall Street Journal corpus to estimate the parameters of an English grammar [3]. The memory requirements of such algorithms are ill-suited for embedded agents exposed to large amounts of training data over long periods of time. If children induced syntax in this manner they would have to memorize a large number of the utterances to which they are exposed, decide at some point to run an algorithm for inducing a grammar from these utterances, and then suddenly have knowledge of the syntax of their native language.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **2002**	2. REPORT TYPE	3. DATES COVERED **00-00-2002 to 00-00-2002**
4. TITLE AND SUBTITLE **Estimating Grammar Parameters using Bounded Memory**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Massachusetts,Department of Computer Science,Amherst,MA,01002**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**14**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

The goal of our work is to develop algorithms for inducing SCFGs from data that have bounded memory requirements and that learn via incremental computation. The former requirement implies that the amount of memory consumed by the algorithm must remain fixed, regardless of the number of strings supplied as input. The latter requirement implies that improvement in the grammar can occur with small amounts of computation and that the quality of the grammar improves monotonically as more computation is allocated to learning. This paper introduces an algorithm called HOLA that satisfies both of these requirements. The novel approach taken by HOLA is justified theoretically, and empirical results show that HOLA performs just as well as the Inside-Outside algorithm in estimating the parameters of SCFGs from data despite the fact that it has access to a bounded amount of information.

2 Background

Following Hopcroft and Ullman [4], a *context-free grammar* (CFG) is a four-tuple $G = (N, \Sigma, P, S)$ where N is a finite set of non-terminals, Σ is a finite set of terminals, P is a finite set of productions or rules, and $S \in N$ is the start symbol. N and Σ are disjoint. Elements of P are of the form $X \rightarrow \alpha$ where $X \in N$ and $\alpha \in (N \cup \Sigma)^*$. The language accepted by G, denoted $L(G)$, is a subset of Σ^*. A grammar is said to be *ambiguous* if for some string $w \in L(G)$ there is more than one way to derive w from S.

A stochastic context-free grammar is a CFG where each production is augmented with a probability. The probability associated with production $X \rightarrow \alpha$ is denoted $p(X \rightarrow \alpha)$. The probabilities of all the productions that expand any given non-terminal must sum to one. The CFG underlying a SCFG is called the SCFG's *structure*, and the probabilities are called its *parameters*. The parameters of a SCFG are denoted Θ. SCFGs define a probability distribution over strings. The probability of a string given a SCFG is the sum over each derivation of the string of the product of the probabilities of the productions used in the derivation.

Given the structure of an unambiguous SCFG it is easy to determine the maximum likelihood parameters for a given training set, i.e. those parameters that maximize the probability of the data given the grammar. Let D be a derivation of some string in the training data and let $c(X \rightarrow \alpha|D)$ be the number of times that production $X \rightarrow \alpha$ occurs in D. The maximum likelihood estimate of a production's probability is as follows:

$$\hat{p}(X \rightarrow \alpha) = \frac{\sum_D c(X \rightarrow \alpha|D)}{\sum_D \sum_{X \rightarrow \beta} c(X \rightarrow \beta|D)}$$

When a grammar is ambiguous there may be many derivations for a given string in the training data and there is no way to know which one was actually used to generate the string. Strings are observable but the actual derivation used to generate a string is hidden. The Inside-Outside algorithm [5, 6] uses Expectation Maximization [7] to solve this hidden data problem. In the expectation step, a weighted sum is computed for each production of the number of times it occurs in the derivations of strings in the training data, with derivation probabilities serving as the weights:

$$\hat{c}(X \rightarrow \alpha) = \frac{\sum_D p(D|G) c(X \rightarrow \alpha|D)}{\sum_D p(D|G)}$$

In the maximization step, these expected counts are used to compute new parameter estimates:

$$\hat{p}(X \to \alpha) = \frac{\hat{c}(X \to \alpha)}{\sum_{X \to \beta} \hat{c}(X \to \beta)}$$

The Inside-Outside algorithm is the gold standard for accuracy of parameter estimates. Other algorithms have been devised for estimating the parameters of SCFGs, such as HOLA, that address limitations of Inside-Outside. But no algorithm has been shown to do consistently better with respect to parameter estimation.

Two approaches that are especially relevant to the research described herein are Neal and Hinton's incremental EM [8] and Boyen and Koller's online EM [9]. The idea behind incremental EM is to speed the convergence of standard EM by running a complete M step after the expected value of each hidden variable is computed, corresponding to a single data item, rather than waiting until the expected values of all hidden variables are computed. Doing so makes information available to the M step more quickly and is shown empirically to speed convergence. That is, incremental EM requires fewer passes through the data than standard EM. The algorithm can be used in an online setting by repeatedly obtaining a new data item, running a partial E step, and discarding the item. However, this greatly increases the total number of data items that must be observed and may not be practical when large amounts of data are required for batch parameter estimation. As previously noted, accurately estimating the parameters of SCFGs often requires large amounts of training data, thereby making incremental EM less attractive.

Boyen and Koller's online EM is based on Neal and Hinton's incremental EM and therefore shares its shortcomings with respect to SCFG parameter estimation. In addition, online EM was applied to parameter learning in dynamic Bayesian networks, a representation that admitted effective belief state approximations, and it is unclear whether the approach is feasible for SCFGs as well.

3 Motivation

The number of times a grammar's productions occur in derivations of strings in the training data plays an important role in parameter estimation. For unambiguous grammars these counts are sufficient for recovering the maximum likelihood parameter estimates. For ambiguous grammars the Inside-Outside algorithm weights the counts by derivation probabilities, a computation that requires storage linear in the size of the training data.

The idea behind HOLA is to use unweighted counts to drive the search for parameters, regardless of whether the grammar is ambiguous or unambiguous. The counts are a function of two things – the structure of the grammar and the training data. The parameters of the learned grammar do not enter into their computation. However, because the training data are sampled according to the distribution over strings defined by the target grammar, the parameters of that grammar do affect the counts. HOLA attempts to find a set of parameters that, given a fixed structure, will generate strings that yield the same (or similar) counts as the training data. Because HOLA keeps a counter for each production in the grammar rather than a set of derivations for the strings in the training data, its memory requirements are linear in the size of the grammar regardless of the size of the training corpus.

The natural way to formulate the search for a set of parameters is in terms of gradient descent. Doing so requires a function that maps from grammars (both structure and parameters) and counts to an error term that indicates how similar the counts are to those that would result from sampling from the grammar. Taking the partial derivative of this function with respect to the parameters of the grammar would make it possible to perform gradient descent in parameter space. The main result of this section is a proof that such a function is not computable and must therefore be approximated.

Given a set of counts, C_1, and a grammar, G, we want to compute the counts, C_2, that would result from sampling from G so that C_1 and C_2 may be compared.

Definition 1. *Let $\phi(X \to \alpha, G)$ be a function that computes the expected number of times production $X \to \alpha$ will occur in the derivation(s) of a string in $L(G)$ sampled according to the distribution over strings defined by stochastic context-free grammar G:*

$$\phi(X \to \alpha, G) = \sum_{s \in L(G)} p(s|G) \left(\sum_{D \text{ of } s} c(X \to \alpha | D) \right)$$

The following lemma will be useful in proving the main theoretical result of this section. It says that for any stochastic context-free grammar G it is possible to create a new grammar G' that has certain desirable properties.

Lemma 1. *Let $G = (N, \Sigma, P, S)$ be a SCFG. Create grammar $G' = (N', \Sigma', P', S')$ from G as follows. Let $N' = N \cup S'$ where $S' \notin N$ and S' is the start symbol of G'. Let $\Sigma' = \Sigma$ and let $P' = P \cup S' \to S$ where $p(S' \to S) = 1$. The following are true:*

(1) $L(G') = L(G)$
(2) $p(w|G') = p(w|G)$ for all $w \in L(G)$
(3) $c(S' \to S|D) = 1$ for any valid derivation D

Proof: By construction, every derivation of a string in $L(G')$ starts by expanding S' to S, where S is the start symbol of G. Therefore, any string that can be derived from S can be derived from S'. Because the productions of G' are identical to those of G except for the one involving S', the derivation(s) of w from S and S' will be identical except for the initial application of $S' \to S$ in the latter case. Because the derivation(s) are the same (after generating S in G') for the two grammars and because $p(S' \to S) = 1$ the probabilities of the strings will be the same. □

Now we are in a position to prove the following theorem.

Theorem 1. *The function ϕ is not computable for an arbitrary production in an arbitrary stochastic context-free grammar.*

Proof: Suppose that ϕ is computable. Let G' be the grammar constructed as described in Lemma 1 for some stochastic context-free grammar G. The construction of G' ensures that $c(S' \to S) = 1$ for every derivation. Consider $\phi(S' \to S, G')$, if G is unambiguous

Θ	*observed*	*norm*	*sample*	*norm*
$S \rightarrow A$ [0.5]	4	0.57	3	0.60
$S \rightarrow B$ [0.5]	3	0.43	2	0.40
$A \rightarrow y$ [0.5]	1	0.25	1	0.33
$A \rightarrow z$ [0.5]	3	0.75	2	0.67
$B \rightarrow z$ [1.0]	3	1.00	2	1.00

Fig. 1. A grammar that generates the language $\{y\, z\}$. Both observed and normalized counts are provided for a bag of strings containing one y and three z's.

then so is G', in which case the inner sum in Definition 1 is one for all strings in $L(G')$ and we have the following:

$$\begin{aligned}\phi(S' \rightarrow S, G') &= \sum_{s \in L(G')} p(s|G') \\ &= 1\end{aligned}$$

If G is ambiguous then there is more than one derivation for some string in $L(G)$ and thus more than one derivation for some string in $L(G')$, in which case the inner sum in Definition 1 is greater than one for that string and $\phi(S' \rightarrow S, G') > 1$. That is, we can use the value of $\phi(S' \rightarrow S, G')$ to decide whether or not G is ambiguous. However, it is undecidable whether an arbitrary CFG is ambiguous [4]. This is a contradiction, so ϕ is not computable. □

The import of Theorem 1 is that we cannot hope to perform gradient descent in parameter space analytically. As described in the next section, HOLA uses sampling to overcome this hurdle.

4 Algorithm Description

This section outlines the HOLA algorithm, gives examples of its execution, and discusses enhancements and improvements. In contrast to other learning algorithms, HOLA does not use the observation (i.e., training) data directly to estimate grammar parameters. Rather, learning is done indirectly by finding parameters that generate strings similar to the those observed. To this end, HOLA exploits the generative nature of grammars as a means for learning.

The HOLA algorithm is given in Figure 2. HOLA attempts to recover the parameters of the grammar generating the observation data. We call this the *target* grammar. The structure of the target grammar is given to the algorithm, but the initial parameters are set by random assignment or pre-training. We call the structure and current parameter estimates the *learning* grammar. Given a learning grammar G and a set of strings S generated from the target grammar, HOLA learns a set of parameters that generate strings statistically equivalent to the observed data.

First, HOLA finds the derivation of each string in S with respect to the grammar. This process is called parsing and occurs in the HOLACOUNT subroutine of the algorithm.

```
HOLA(scfg,strings)
1.  HOLACOUNT(scfg,strings)
2.  UNLESS STOPPINGCRITERION(scfg)
3.      HOLAITERATION(grammar)

HOLACOUNT(scfg,strings)
1.  derivations ← PARSE(scfg,S)
2.  FOREACH d in derivations
3.      FOREACH r in scfg.rules
4.          r.observed ← r.observed + COUNT(r, d)
5.  NORMALIZEOBSERVEDCOUNTS(scfg)

HOLAITERATION(scfg)
1.  sampleStrings ← SAMPLE(scfg)
1.  derivations ← PARSE(scfg,sampleStrings)
2.  FOREACH d in derivations
3.      FOREACH r in scfg.rules
4.          r.sample ← r.sample + COUNT(r, d)
5.  NORMALIZESAMPLECOUNTS(scfg)
6.  UPDATEPARAMETERS(scfg)
```

Fig. 2. The HOLA algorithm.

Parsing is a function of the grammar structure, not the parameters. When the grammar is ambiguous, multiple derivations may exist for a single string. For example, consider the grammar in Figure 1 and the set of strings $\{y\,z\,z\,z\}$. The string y has a single derivation, $S \rightarrow A \rightarrow y$, but z has two derivations, $S \rightarrow A \rightarrow z$ and $S \rightarrow B \rightarrow z$. Each possible derivation indicates what rules were used in generating the string. HOLA finds the total occurrences of each rule in all the derivations, records them, and then disposes of them. We call these *observed* counts since they come from the observed data. Because HOLA searches for parameter estimates that produce strings with counts similar to those observed, we need a general way to compare counts. For comparison, HOLA normalizes the counts with respect to rules with the same left-hand-side. Only the normalized count for each rule is stored. The observed counts are not updated, but stay fixed throughout the rest of HOLA's execution. Both the observed and normalized counts for the data discussed above are given in Figure 1.

Next, HOLA iterates through a generate and update cycle until a stopping criterion is met. This corresponds to the HOLAITERATION subroutine in the algorithm. This procedure is nearly identical to HOLACOUNT except for two differences. First, the observed data is replaced with a small sample of strings generated from the grammar. This sample reflects the current parameter estimates. For example, generating a sample of size three from the grammar in Figure 1 will probably result in two z's and one y. Second, counts taken from the sample are stored separately from the observed counts. At the end of the generation phase, each rule r has two counts $r.observed$ and $r.sample$. The pairwise similarity of these counts indicates the similarity in the current parameter estimates and the target parameters. HOLA updates each rule according to these differences:

$$p(r) = p(r) * (1 + \alpha * (r.observed - r.sample))$$

Note that when the sample counts are smaller than the observed counts, the rule probability increases. When the sample counts are larger, the rule probability decreases. The change in parameter estimates potentially changes the strings we would expect to see when generating a sample from the grammar during the next iteration. These changes in turn move the parameters toward more likely estimates. The step-size parameter α helps learning narrow in on the correct parameter estimates. However, since each iteration generates a set of strings, convergence to maximum likelihood estimates probably does not happen because of sample variance.

5 Experiments

This section shows empirically that HOLA learns good parameter estimates using bounded memory. We performed three experiments: two on unambiguous grammars generating English phrases and palindromes and one on a small ambiguous grammar. In each experiment we fixed the structure of the target grammar and conducted 50 independent trials, randomly generating the target parameters in each case. A trial consists of generating 1000 strings of observation data from the target grammar. Next a learning grammar is created by taking the structure of the target grammar and reinitializing it with new random parameters. Finally, copies of the learning grammar are handed along with the observation data to both HOLA and the Inside-Outside algorithm.

The Inside-Outside algorithm is known to converge to a set of parameters that locally maximize the likelihood of the data. We show HOLA performs comparably to the Inside-Outside algorithm even though it uses less information and requires only bounded memory. This evaluation of the learned parameter estimates is accomplished by finding the log-likelihood of the data given the grammar and the learned parameters.

5.1 English Phrases

We used the English phrase grammar taken from Cook, Rosenfeld and Aronson [10] in Figure 3 in our first experiment. This grammar is unambiguous and does not contain any recursive rules, however, it is comparable in size to other grammars used in the literature for grammatical inference (e.g., [11]). We ran HOLA for 100 iterations using a sample size of 100 and decreasing the step-size parameter by 10% every 10 iterations. We allowed the Inside-Outside algorithm to run until convergence. HOLA performed well in comparison to the Inside-Outside algorithm in all trials. Figure 4 gives the percentage difference between HOLA and the Inside-Outside algorithm with respect to the log-likelihood of the data given the learned parameters. In all trials the difference in performance was less than one percent; in over half the trials, the difference was less than two-tenths of one percent. The mean difference was 0.166 percent, the variance only 0.017 percent. In most cases the total difference in true log-likelihood was fractional. We expect the differences will converge to zero once a suitable method for reducing

$$
\begin{array}{lccc}
S \rightarrow & I & am & A \\
S \rightarrow & he & T & \\
S \rightarrow & she & T & \\
S \rightarrow & it & T & \\
S \rightarrow & they & V & \\
S \rightarrow & you & V & \\
S \rightarrow & we & V & \\
S \rightarrow & this & C & \\
S \rightarrow & that & C & \\
T \rightarrow & is & A & \\
V \rightarrow & are & A & \\
Z \rightarrow & man & & \\
Z \rightarrow & woman & & \\
A \rightarrow & there & & \\
A \rightarrow & here & & \\
C \rightarrow & is & a & Z \\
C \rightarrow & Z & T &
\end{array}
$$

Fig. 3. A grammar generating English strings from Book, Rosenfeld and Aronson

sample variance is incorporated into HOLA. Empirically, though, HOLA tends to find parameter estimates strikingly similar to those found by the Inside-Outside algorithm.

HOLA is also robust to differences in initial parameter settings. Applying linear regression to the trials plotted as a function of performance difference and sorted initial loglikelihood results in a near horizontal line (see Figure 5 with $r^2 = 0.03$. This means little correlation exists between HOLA's performance and the initial log-likelihood.

Figure 6 shows the learning curve over 100 iterations for trial 1. Note that by iteration 40, HOLA has settled in on good parameter estimates. Each subsequent iteration walks locally around the maximum likelihood probably due to sample variance.

5.2 Palindromes

The second experiment involved the palindrome-generating grammar in Figure 7. This grammar is unambiguous and contains two self-referential rules. HOLA ran for 300 iterations while decreasing the step-size by 5% every 10 trials. The results in Figure 8 show that in all trials, HOLA's performance differs from the Inside-Outside algorithm by less than one-half of one percent. In three quarters of the trials, the performance difference was less than one-tenth of one percent. The mean percentage difference is 0.07, the variance 0.009. Like the first experiment, HOLA learns parameter estimates only fractions away from those learned by Inside-Outside algorithm.

5.3 Ambiguous Grammars

Our final experiment used the simple ambiguous grammar discussed previously in Figure 1. We ran HOLA for 100 iterations with the step-size parameter decreasing every 10 iterations by 5%. Like in the other experiments, HOLA performs almost identically to

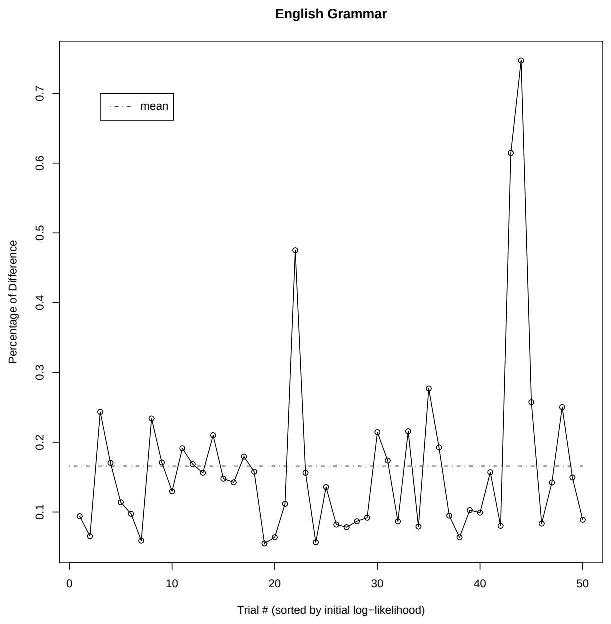

Fig. 4. The percentage difference in log-likelihood between HOLA and the Inside-Outside algorithm for 50 trials using the English phrase grammar.

the Inside-Outside algorithm. In all the trials, save three, the percentage difference in log-likelihood was less than half a percent. In an overwhelming majority of cases, the difference was less than one-tenth of one percent. The mean difference in percentage was 0.17, however, if we remove the three outliers the mean falls to 0.03. The variance was 0.34, however, removing the outliers significantly reduces it to 0.002.

The higher difference in the three outliers occurs because learning isn't finished. For example, consider the farthest outlier, trial 2. Here, the negative log-likelihood after 100 iterations is around 192. The local maximum likelihood is 186.56. If we allow learning to continue for 200 more iterations, HOLA finds better parameter estimates resulting in a negative log-likelihood of 186.82 – only fractionally different from those found by the Inside-Outside algorithm.

6 Discussion

Consider again the example grammar given in Figure 10. We know every SCFG defines a probability distribution over the language of the grammar. In this case the distribution is $p(y) = .3$ and $p(z) = .7$. Said differently, if we generate 10 sentences from our grammar, we expect to see three y's and seven z's. In fact, the bag of strings containing three y's and seven z's is the smallest corpus completely representative of the probability distribution provided by Θ. That said, note that the observed counts of each rule, when

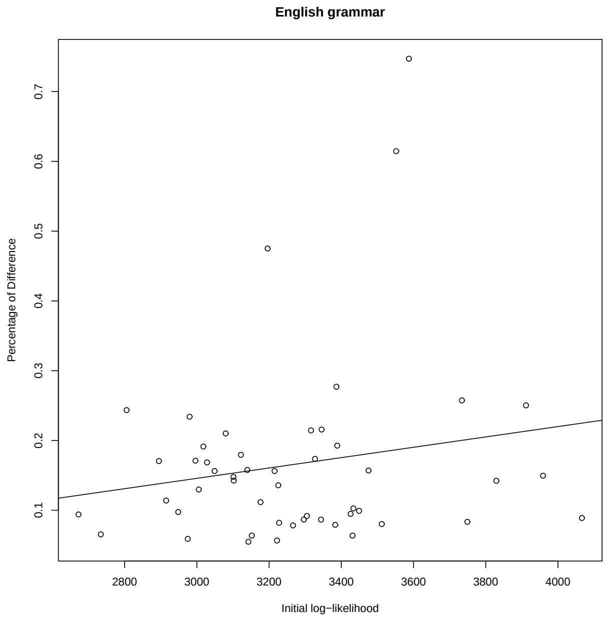

Fig. 5. Percentage of difference in performance versus sorted initial log-likelihood on the English phrase grammar. The line is a linear fit of the data.

suitably normalized, are poor estimators of the original parameters. This is because the grammar is ambiguous. Furthermore, setting Θ to the normalized counts yields a completely different probability distribution over the language; $p(y) \approx .18$ and $p(z) \approx .82$. But, recall that HOLA does not use the counts directly, but rather attempts to find parameter estimates where the sample normalized counts are equivalent to the observed normalized counts—parameters that result in the *observed* probability distribution over the language.

If we let $p_1 = p(S \rightarrow A)$ and $p_2 = p(A \rightarrow y)$ then $1 - p_1 = p(S \rightarrow B)$ and $1 - p_2 = p(A \rightarrow z)$. Any parameterization $p_1, p_2 \in [0, 1]$ satisfying $p_1 p_2 = 0.30$ results in a probability distribution over the language where y's occur 30% of the time and z's 70%. Clearly these parameters may vary significantly from those in Θ. However, from a generative view, they are good estimators since the expected output is equivalent to the generating grammar.

One natural question is: *Does a parameterization Θ' exist for a grammar such that the normalized counts are the same but the probability distribution over the language is different?* For the grammar at hand the answer is 'no.' The only way y can be generated is through an application of $A \rightarrow y$, so we know the normalized count for $A \rightarrow y$ is $p(y)$. This means the normalized count for $A \rightarrow z$ is $p(z) = 1 - p(y)$. Since $S \rightarrow B$ is counted with the same frequency as $A \rightarrow z$, its normalized count is $p(z)/(1.0 + p(z))$; the 1.0 in the denominator is added because $S \rightarrow A$ can derive the entire language.

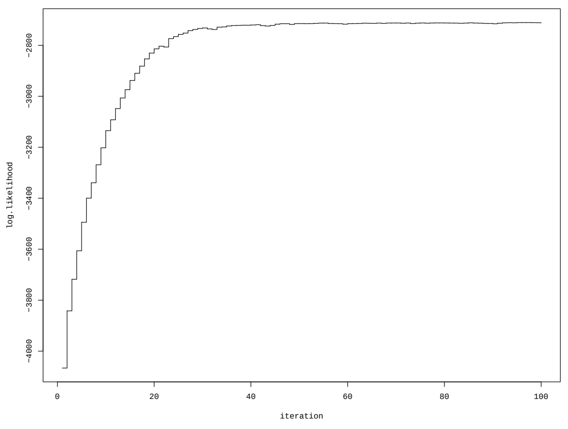

Fig. 6. HOLA's learning curve in trial 1 of the English phrase grammar.

$$\begin{array}{lll}
S & \rightarrow & A\ S\ A \\
S & \rightarrow & B\ S\ B \\
S & \rightarrow & A\ A \\
S & \rightarrow & B\ B \\
S & \rightarrow & A \\
S & \rightarrow & B \\
A & \rightarrow & y \\
B & \rightarrow & z
\end{array}$$

Fig. 7. A grammar generating palindromes over the alphabet $\{y\ z\}$

This means $S \rightarrow A$ has a normalized count of $1/(2 - p(y))$. It's clear for this grammar that the probability distribution over the language corresponds linearly with the normalized counts. This means fixing the counts results in only one possible probability distribution over the language. To the best of our knowledge, whether this is true for all stochastic context-free grammars is still an open question. We suspect that grammars exist where multiple parameter estimates lead to different probability distributions over the language while still resulting in identical rule counts, but these estimates locally maximize the likelihood of the data.

7 Conclusion

The HOLA algorithm raises and addresses some interesting theoretical and empirical questions. First, it incrementally learns likely parameter estimates of stochastic context-free grammars using bounded space. Such algorithms are developmentally more plausible and applicable in domains where large amounts of data are encountered and processed over long periods of time. Second HOLA shows that using the generative nature of grammars helps capriole the hurdle of analytically determing rule counts. At the same time, sample variance hinders convergence but we're confident that future work will address and solve this problem. Still, empricial evidence shows that the Inside-Outside

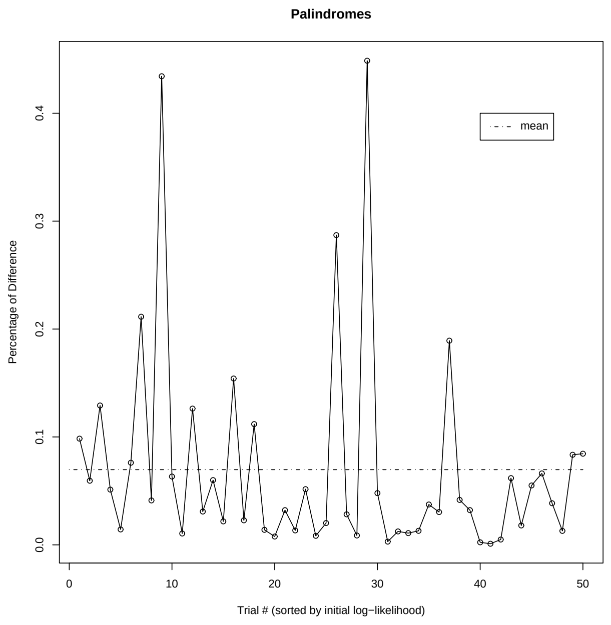

Fig. 8. The percentage difference in log-likelihood between HOLA and the Inside-Outside algorithm for 50 trials with the palindrome grammar.

algorithm, known to converge to parameters that are locally maximum, performs only fractionally better than HOLA. Finally, we discussed the quality of the learned estimates, specifically asking where in parameter space estimates that produce counts similar to the data lie. While emprically the estimates move toward local maximum likelihood locations, in the future we hope to show theoretical proof of such convergence.

8 Acknowledgements

Mark Johnson supplied a nice implementation of the inside-outside algorithm.

This research is supported by DARPA/USASMDC under contract number DASG60-99-C-0074. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA/USASMDC or the U.S. Government.

References

1. Sakakibara, Y., Brown, M., Highey, R., Mian, I.S., Sjolander, K., Haussler, D.: Stochastic context-free grammars for tRNA modeling. Nucleic Acids Research **22** (1994) 5112–5120

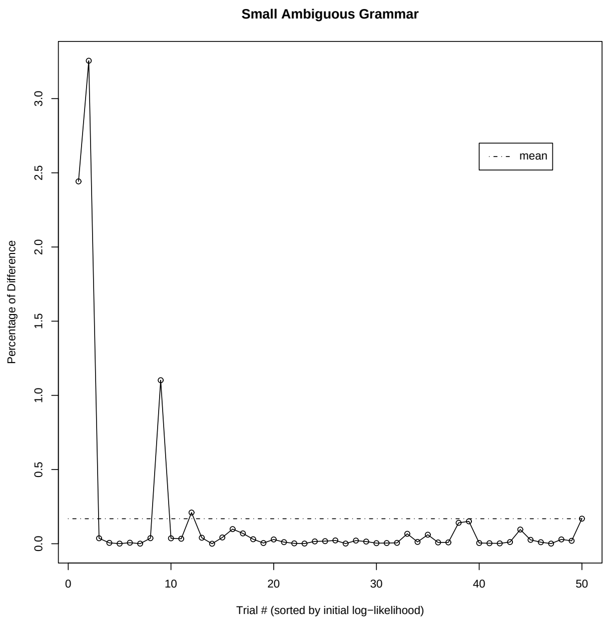

Fig. 9. The percentage difference in log-likelihood between HOLA and the Inside-Outside algorithm for 50 trials with an ambiguous grammar.

2. Jurafsky, D., Wooters, C., Segal, J., Stolcke, A., Fosler, E., Tajchman, G., Morgan, N.: Using a stochastic context-free grammar as a language model for speech recognition. In: Proceedings of ICASSP. (1995) 189–192
3. Schabes, Y., Roth, M., Osborne, R.: Parsing the Wall Street Journal with the inside-outside algorithm. In: Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics. (1993) 341–346
4. Hopcroft, J.E., Ullman, J.D.: Introductin to Automata Theory, Languages, and Computation. Addison Wesley (1979)
5. Lari, K., Young, S.J.: The estimation of stochastic context-free grammars using the inside-outiside algorithm. Computer Speech and Language **4** (1990) 35–56
6. Lari, K., Young, S.J.: Applications of stochastic context-free grammars using the inside-outside algorithm. Computer Speech and Language **5** (1991) 237–257

	Θ	*observed*	*normalized*
$S \rightarrow A$	[0.6]	$10/17$	0.59
$S \rightarrow B$	[0.4]	$7/17$	0.41
$A \rightarrow y$	[0.5]	$3/10$	0.30
$A \rightarrow z$	[0.5]	$7/10$	0.70
$B \rightarrow z$	[1.0]	$7/7$	1.0

Fig. 10. A grammar that generates the language $\{y\, z\}$

7. Dempster, N.M., Laird, A.P., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B **39** (1977) 185–197
8. Neal, R.M., Hinton, G.E.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M.I., ed.: Learning in Graphical Models, Kluwer Academic (1998)
9. Boyen, X., Koller, D.: Approximate learning of dynamic models. In: Neural Information Processing Systems. (1998)
10. Cook, C.M., Rosenfeld, A., Aronson, A.: Grammatical inference by hill climbing. Informational Sciences **10** (1976) 59–80
11. Stolcke, A., Omohundro, S.: Inducing probabilistic grammars by bayesian model merging. In Carrasco, R.C., Oncina, J., eds.: Grammatical Inference and Applications, Berlin, Heidelberg, Springer (1994) 106–118